METHOD AND SYSTEM FOR VIRTUALIZING LOGIC BETWEEN DISPARATE SYSTEMS

BACKGROUND OF THE INVENTION

The invention disclosed herein relates generally to methods for providing access to computer system logic. More particularly, the present invention relates to a method and system for virtualizing the logic of computer systems by providing access to the logic of a first computer system at a second computer system such that the logic of the first system appears native to the second system.

Companies today often have computing infrastructures comprising many distinct computer systems arising from such situations as where newly purchased systems must be integrated with older systems or where organizations with different systems merge. The result is that various portions of computer logic, e.g., computer programs and procedures, needed by a company often exist on several different systems. For example, a company may have logic in SAP, stored procedures in IBM DB/2, and applications in Lotus Domino.

Having a company's computing logic on several different systems is undesirable since the personnel who must use that logic must become familiar with each different system. As a result, there have been attempts to integrate different systems so that, through a single front-end system, a user may be able to access the logic of several different systems. The logic of the different systems becomes "virtualized" in that the logic of the different systems appears to be and is usable in the same manner as the logic of the front-end system. The user would then only need to be familiar with the operation of the front-end system.

To date, efforts to integrate, or "virtualize", the logic of disparate systems have focused on either (1) rewriting the different pieces of logic into a single system or (2) creating

complex code to connect the logic at the separate systems. Both methods have significant drawbacks for the Information Technology development staff of the company in the form of development time and expenditure as well as maintenance costs resulting from the introduction of program bugs.

There is thus a need for a method for virtualizing the logic of disparate systems using distinct programming models and data types that does not require the rewriting or creation of new code on the part of the development staff of the company owning the disparate systems.

BRIEF SUMMARY OF THE INVENTION

It is an object of the present invention to provide a method and system for virtualizing the logic of disparate computer systems that avoids the drawbacks described above.

It is another object of the present invention to perform virtualization of logic without creating new programming code.

It is another object of the present invention to provide a method for virtualizing the logic of disparate computer systems that requires minimal development time.

It is another object of the present invention to provide a method for virtualizing the logic of disparate computer systems that requires only minimal computer expertise.

The above and other objects are achieved by a method for providing access to the logic units of a first computer system at a second computer system. Computer system here is used broadly to mean computer hardware and software or computer software only. The first system has one or more logic units that are externally invocable. The second system issues searches for invocable logic units and commands to invoke logic units that may be captured externally. The method involves capturing a search for invocable logic units issued from the

second system, returning a list of one or more externally invocable logic units of the first system as a result for the captured search, capturing a command to invoke a logic unit issued from the second system, and if the logic unit identified in the captured command is a listed logic unit, causing the first system to invoke the identified logic unit, receiving the results of the invocation of the identified logic unit from the first system, and returning the results to the second system.

The first system, which may be called the external system, may have logic, in the form of distinct and independently executable logic units, that may be invoked by an outside system. The second system, which may be called the front-end system, may have hooks that enable an outside system to capture the searches for invocable logic units and the commands to invoke logic units that the second system issues. Capture means any action or technique through which the execution of computer logic may be directed to a particular target system. Command is used here to mean any communication requesting that a task be performed.

The method may begin with an initialization of the system of the invention. This may begin with an administrator providing to an Administration Tool (a) the identity of one or more external systems containing logic to be virtualized and (b) the identity of the one or more logic units at the external system(s) to be virtualized. The identity of the front-end system for which logic will be virtualized must also be known to the Administration Tool. This information may be provided to the Administration Tool by an administrator. Alternatively, the identity of the front-end system may be predetermined so that it need not be provided. With regard to (b), the administrator may directly provide the names of the logic units to be virtualized or may provide a search pattern or any criteria for guiding a search. If a search pattern is provided, the external systems of (a) will return the identities of their logic units that match the pattern.

The above information may be provided to the Administration Tool through any known technique for providing information to a computer system, such as filling in a form at a graphical user interface, placing the information in a data file and providing the file to the computer system, etc. The information provided to and received by the Administration Tool is forwarded to and stored at the system of the present invention.

Also as part of the initialization, the system of the present invention is associated with the front-end system so as to allow the system of the present invention to capture the front-end system's search for invocable logic units and capture the front-end system's command to invoke logic units. This association may be done by any known technique for linking computer systems, such as registering the system of the present invention as a client of the front-end system's hooks.

After initialization, the system of the present invention waits for an end user of the front-end system to make a request to invoke logic. The front-end system handles such a request in two parts. First, it issues a search for all logic units that may be invoked and presents the identities of the invocable logic units to the end user. These identities may be returned in a list or in any other useful manner for presenting information. After the end user selects one of the identified logic units, the front-end system issues a command to invoke the selected logic unit.

Due to the previously made association, the system of the present invention captures the search for logic units that may be invoked and provides the stored list of logic units from external systems that are to be virtualized at the front-end system. The system of the present invention also allows the front-end system to search for its internal, or native, logic units that may be invoked. The list of invocable logic units that is presented to the end user then contains logic units native to the front-end system and virtual logic units.

Express Mail No. EL595826225US

Once the end user selects a logic unit from this list to be invoked, then due to the previously made association, the system of the present invention captures the command to invoke the selected logic unit that issues from the front-end system. If the selected logic unit is virtual, the system of the present invention extracts information from the front-end system, such as the data the logic unit is to operate on and, if necessary, parameters for the logic unit. The system of the present invention then converts that information from the format it existed in at the front-end system to a central format at the system of the present invention. The system of the present invention then converts the information to the format of the external system whose logic unit has been selected to be invoked.

The system of the present invention commands the external system whose logic unit has been selected to invoke that logic unit and the system of the present invention also passes the converted information. The external system invokes the selected logic unit and returns the resulting data to the system of the present invention which then converts the received data to the format of the front-end system and returns the converted data to the front-end system. The front-end system then presents the returned data to the end user.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is illustrated in the figures of the accompanying drawings which are meant to be exemplary and not limiting, in which like references are intended to refer to like or corresponding parts, and in which:

Fig. 1 is a block diagram showing the various systems that operate with the system of the present invention;

Fig. 2 is a flow chart showing the operation of the present invention; and

Fig. 3 is a flow chart further showing the operation of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The preferred embodiments of a method, system, and article of manufacture containing software programs in accordance with the present invention is described with reference to the drawings in Figs. 1-3.

Fig. 1 is a block diagram showing the operating environment of a system of the present invention. Front-End System 100 is a computer system that directly communicates with one or more end users each operating an End User Computer System 200. Front-End System 100 contains logic comprising one or more discrete and independently executable units, known hereinafter as "logic units". Various examples of different types of logic units are known in the art, including procedures, agents, routines, objects, etc. Logic units may operate on various datatypes, may accept and use various parameters as input to their execution, and may return data results as output from their execution.

Also, logic units may carry out various types of functions. For example, systems that process business data may store that data in databases and have rules embodied in procedures that operate on the data. An example of a business related logic unit could be a procedure for adding a new customer to a system. The information for a given customer may comprise many datatypes stored at various locations within the system. The "AddNewCustomer" procedure would insure that the correct information was received and that the correct information and datatypes are stored at the correct locations within the system.

Front-End System 100 contains one or more logic units 1 through *n*. The logic units shown for Front-End System 100 are internal to Front-End System 100 and may be called "native" logic units.

Front-End System 100 also includes two hooks. Hooks are well known to those of skill in the art as providing a means through which the execution of a system's logic may be captured and redirected to an outside system.

Front-End System 100 includes a first hook 110 that allows an outside system to capture the execution of the Front-End System 100's search for invocable logic units. When an end user of Front-End System 100 chooses to invoke logic, Front-End System 100 executes a search to identify the logic units that are invocable by a user. If no outside system is linked to hook 110, then this search returns only a list of logic units native to Front-End System 100. The list is then presented to the end user.

However, if an indication has been provided to Front-End System 100 that an outside system is linked to hook 110, then the search for invocable logic units will be captured and redirected to the outside system linked to hook 110. The outside system may then return its own list of invocable logic units to Front-End System 100. Afterwards, execution may continue with the search to identify invocable native logic units. Thus, if an outside system is linked to hook 110, then when an end user of the Front-End System chooses to invoke logic, the Front-End System may present a list of invocable logic units that may include logic units identified by the outside system as well as logic units native to the Front-End System.

Front-End System 100 also includes a second hook 120 that allows an outside system to capture the invocation of logic units at Front-End System 100. As described above, when an end user chooses to invoke logic, a search is executed and a list of invocable logic units

is returned. This list is presented to the end user who then selects one of the logic units to be invoked. If no outside system is linked to hook 120, then the Front-End System issues a command to invoke the selected logic unit itself and presents the results to the end user.

However, if an indication has been provided to Front-End System 100 that an outside system is linked to hook 120, then the command to invoke, including the identity of the logic unit selected to be invoked, is captured and redirected to the outside system linked to hook 120. The outside system may then invoke logic units outside of Front-End System 100 and return the results of the invocation to Front-End System 100. Alternatively, if the captured command to invoke indicates that a native logic unit is selected, then the outside system may simply return to allow the Front-End System to invoke the native logic unit.

Thus, Front-End System 100 may be any computer system that includes (1) a hook that enables an outside system to capture Front-End System's search for invocable logic units and return its own list of invocable logic units in response, and (2) a hook that enables an outside system to capture commands to invoke logic units at Front-End System 100, invoke logic units not native to Front-End System 100, and return the results of the invocation to Front-End System 100. Lotus Domino is an example of a computer system that may serve as Front-End System 100. Other examples of systems that may serve as front-end systems include database programs, groupware programs, etc.

Fig. 1 also shows one or more External Systems 300, which may also be called the back-end systems. Each External System 300 is a distinct system, which may comprise computer hardware and/or computer software, that also contains logic that may be represented as one or more logic units. The logic of each External System 300 includes one or more externally

invocable logic units 1 through *m*, some or all of which are logic units that may be invoked by a system external to that particular back-end system.

Thus, an External System 300 may be any system having one or more logic units that may be cataloged and invoked by an outside system. The term catalog is used here broadly to refer to any technique or method through which an outside system may request and receive from the external system the identities of all logic units of the external system that are externally invocable. Examples of computer systems that may serve as external systems 300 include IBM DB/2, Oracle database systems, Microsoft Access, and Lotus Domino.

The system of the present invention, Integrating System 400, virtualizes selected externally invocable logic units of one or more External Systems 300 for Front-End System 100. Once they are virtualized by Integrating System 400, externally invocable logic units of an External System 300 are presented to and may be invoked by end users of Systems 200 just as logic units native to Front-End System 100 are presented to and may be invoked by end users from Systems 200.

An Administration Tool 500 is used to identify (1) one or more External Systems 300 having logic unit(s) to be virtualized, and (2) one or more externally invocable logic units from the External System(s) 300 that are to be virtualized. The identity of the Front-End System 100 through which the virtual logic units will be presented and invoked must also be known to the Administration Tool. This identity may be provided to the Administration Tool or the identity of Front-End System 100 may be predetermined. Administration Tool 500 includes a user interface through which it obtains the above information from a human operator, e.g., the administrator. This information is stored at Administration Tool 500 until required by the

Integrating System. Administration Tool 500 then passes the information to Integrating System 400, which stores the names of the logic units, 1 through p, to be virtualized in a list 410.

Administration Tool 500 may be a component of the Integrating System, but may also be a separate system itself.

Integrating System 400 also includes a conversion mechanism 420 used for converting the datatypes and types of logic unit parameters of a system outside of Integrating System 400 into central datatypes and parameter types and vice versa. Thus, conversion mechanism 420 provides a "hub-and-spoke" technique, for allowing Integrating System 400 to convert the datatypes and parameter types of a first system into the datatypes and parameter types of a second system by first converting the datatypes and parameter types of the first system into the central datatypes and parameters of conversion mechanism 420 and then converting the central datatypes and parameters to the datatypes and parameters of the second system.

With virtualization, end users at End User Systems 200 that request to invoke logic at Front-End System 100 will be presented with a list of invocable logic units 1 through q, where the list includes the names of native logic units 1 through n from Front-End System 100 and the names of virtualized logic units 1 through p from Integrating System 400. All the logic units on this list, both native and virtual, are presented to the end user as though they were native to and directly invocable through Front-End System 100.

Fig. 2 is a flow chart describing the steps performed by an embodiment of the invention. First the search for invocable logic units issued from Front-End System ("FES") 100 is captured, step 1000. A list of the externally invocable logic units from the External Systems ("ES") 300 that are to be virtualized is returned to FES 100 as a result of the captured search, step 1100. When FES 100 issues a command to invoke a logic unit, that command is captured,

step 1200, and it is determined whether the logic unit identified in the command is one of the logic units from the list of virtual logic units, step 1300. If the identified logic unit is a virtual logic unit, the identified logic unit at the ES 300 is caused to be invoked, step 1400. After the invocation of the identified logic unit is complete, the resulting data from the invocation is received from the ES 300, step 1500. The resulting data is then returned to FES 100, step 1600.

Fig. 3 is a flow chart that further describes how the virtualization of the present invention is accomplished. First the externally invocable logic units of the External Systems 300 that are to be virtualized are identified, step 2000. This identification step may be performed by an Administration Tool 500. At Administration Tool 500, a human administrator provides information on the virtualization, for example, by filling out a form. This information identifies (1) the Front-End System 100 through which the virtual logic units will be presented and invoked (e.g., Lotus Domino or Notes database), (2) one or more External Systems 300 whose logic units are to be virtualized (e.g., Oracle database, IBM DB/2), and (3) the externally invocable logic units from these External Systems 300 that are to be virtualized.

The administrator may identify the logic units to be virtualized by providing their names and their corresponding parameters, if any. Alternatively, a search pattern may be specified and the External System 300 contacted to determine and return a list of externally invocable logic units matching the search pattern and the parameters corresponding to each logic unit. For example, for the logic units to be virtualized, the administrator could specify the search pattern "a*" (the letter "a" and a wildcard). The External System 300 would then return a list of all its externally invocable logic unit having names beginning with the letter "a", and for each returned logic unit, the parameters, if any, corresponding to that logic unit.

It should be noted that Administration Tool 500 may provide for the conversion of the names of logic units into forms more understandable by an end user. For example, External System 300 may have externally invocable logic units having names consisting of a combination of letters and numbers that may have been initially assigned by the developer of the logic and that are unintelligible to an end user. Administration Tool 500 may provide a mapping whereby names from an external system are mapped to names describing the function of the logic unit, such as "AddCustomer", "DeleteCustomer", etc.

Administration Tool 500 then passes the Front-End System information, the External System information, and a list of logic units to be virtualized, including any corresponding parameters, to Integrating System 400. The names of the logic units to be virtualized, and any corresponding parameters, are stored in list 410 at Integrating System ("IS") 400, step 2100.

In step 2200, IS 400 is associated with Front-End System ("FES") 100 so that FES 100's (1) search for invocable logic units and (2) commands to invoke logic units are captured and redirected to IS 400. This association may be accomplished in various ways depending on the specific system being used as FES 100. For example, Lotus Domino supports hooks for (1) and (2) through Domino Extension Manager. An outside system may register with Domino to be a client of these hooks by adding an appropriate entry in an initialization file, e.g.., an ".ini" file, and providing the code for IS 400, such as through a "shared library" or a ".dll" file.

Although the association step is presented at step 2200, it should be appreciated by one of ordinary skill in the art that this association step may occur prior to steps 2000 and 2100.

Express Mail No. EL595826225US

After initialization is complete, IS 400 waits for a request to search for invocable logic units to be received at FES 100, step 2300. Due to the association created between FES 100 and IS 400, all such requests received at FES 100 are captured and redirected to IS 400, step 2400. IS 400 returns to FES 100 the list previously stored at IS 400 identifying the logic units to be virtualized, along with any corresponding parameters, and then IS 400 allows FES 100 to continue as normal to search for its own list of invocable native logic units, step 2500.

The names of the invocable logic units, both native to FES 100 and virtual to FES 100 via IS 400, are presented to an end user at an End User System ("EUS") 200, for example, in list format. If any of the logic units require parameters that must be provided by the user, those parameters are presented to the user as well. The end user then selects one of the listed logic units to invoke and identifies the data at FES 100 on which the selected logic unit is to operate. If the selected logic unit requires parameters from the user, the user must input that information as well. FES 100 then issues a command to invoke the logic unit selected from the list.

Due to the association created between FES 100 and IS 400, all commands to invoke logic units at FES 100 are captured and redirected to IS 400, step 2600. IS 400 then screens the command to determine if the logic unit selected to be invoked is native or virtual to FES 100, step 2700.

IS 400 may determine if the selected logic unit is virtual in a number of ways. For example, the name of the selected logic unit may be compared with the list 410 containing the names of the logic units to be virtualized to determine if the selected logic unit appears on the this. Alternatively, a naming convention may be used where names having one format are native and names having a different format are virtual. Thus, IS 400 may determine whether the selected logic unit is native or virtual simply from an examination of the logic unit's name.

If the selected logic unit is native, then IS 400 returns to FES 100 to allow it to invoke the selected native logic unit normally. If the selected logic unit is virtual, then IS 400 extracts information from the captured command and FES 100, step 2800. Specifically, IS 400 extracts the name of the virtual function to be invoked from the captured command and extracts the data to be operated on from FES 100. Alternatively, IS 400 may extract from the captured command information that describes how the data to be operated on may be obtained from FES 100, e.g., the identities and locations of target records. If parameters are specified by the end user, IS 400 extracts those from the captured command and/or FES 100. In addition to user specified parameters, the External System 300 of the selected logic unit may have specified parameters to IS 400 via Administration Tool 500 that were not presented to the end user. If so, IS 400 extracts the data representing these parameters from FES 100 as well.

It should be noted that some systems may not directly support parameters and therefore the data to be operated on is not directly provided to IS 400 through the hook. However, the above described extraction of data and parameters may still be accomplished. For example, to facilitate capturing and parameter passing in Lotus Domino, the integrating system creates two Domino agents, one that is normally visible to the end user (linked to the front-end system, here Domino, through an end user system) and one that remains hidden at the Domino server. This convention is necessary because with Domino, a hook can only be triggered on the server, and end-users cannot directly trigger the execution of an agent on the server. When requesting to invoke a logic unit, the end-user executes the visible agent and may select one or more target records on which the selected logic unit is to operate. The visible agent then executes the hidden agent on the server which triggers the hook. The IS 400 then captures this command to invoke the selected logic unit. If there are parameters, the identities of the target

records are temporarily stored in the hidden agent so that once IS 400 is triggered via the hook, it is able to extract these identities and use them to access the target records at Domino, extract the parameters from those records, and invoke the selected logic unit at the corresponding external system.

In step 2900, IS 400 converts the extracted information from a format associated with FES 100, which may be the format in which the information was received or the "source" format, to a format expected by the ES 300 corresponding to the virtual logic unit selected to be invoked, which may be called the "target" format. As described above, in an embodiment of the invention, IS 400 uses a conversion mechanism 420 that employs a "hub-and-spoke" technique for this conversion. Conversion mechanism 420 defines a central concept of procedures, parameters, and datatypes, all of which may be called the "central" format. Conversion mechanism 420 converts the extracted information from the source format to the central format and then from the central format to the target format. The types of data that may be converted from one format to another include numbers, datetimes, text, binary large objects ("BLOB"), etc.

In step 3000, IS 400 instructs the ES 300 owning the externally invocable logic unit corresponding to the virtual logic unit selected to invoke that logic unit and IS 400 also passes to ES 300 the converted data to be operated on and parameters, if any. Note that IS 400 may instruct the ES 300 to invoke the selected logic unit more than once depending on the data to be operated on and the parameters. For example, if the data to be operated on is a plurality of records, then IS 400 will instruct ES 300 to invoke the selected logic unit once for each record.

After the execution of the logic unit completes, the ES 300 returns the resulting data from the execution to IS 400 which converts this data from the ES 300 format to the central

format to the FES 100 format in a manner similar to that previously described, step 3100. IS 400 then returns the converted, resulting data to FES 100, step 3200.

While the invention has been described and illustrated in connection with preferred embodiments, many variations and modifications as will be evident to those skilled in this art may be made without departing from the spirit and scope of the invention, and the invention is thus not to be limited to the precise details of methodology or construction set forth above as such variations and modification are intended to be included within the scope of the invention.